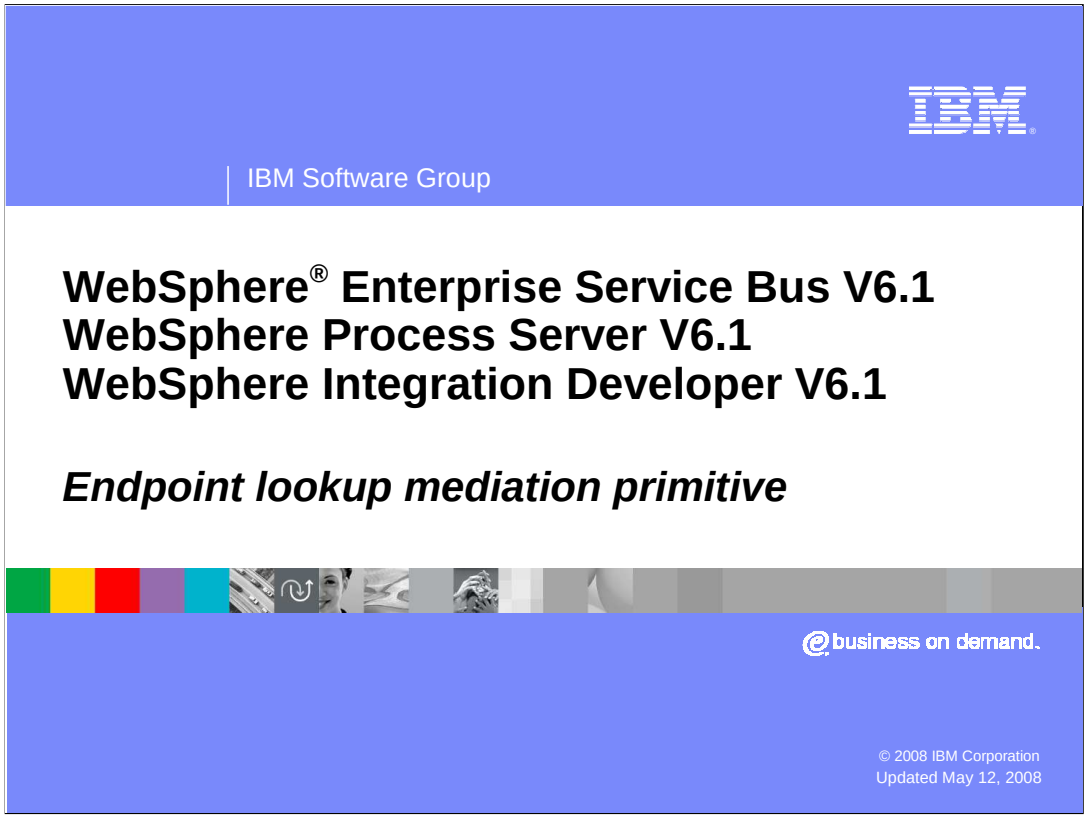


This presentation provides a detailed look at the endpoint lookup mediation primitive.

Goals

- Understand the endpoint lookup mediation primitive


Endpoint lookup

  - Overview of function
  - Use of terminals
  - Definition of properties
  - Service message object (SMO) usage
  - Service registry cache
  - Server administration of the registry
  - Configuration of the registry
  - Error handling
  - Problem determination
  - Example usage

The goal of this presentation is to provide you with a full understanding of the endpoint lookup mediation primitive.

The presentation assumes that you are already familiar with the material presented in the **Mediation Primitive Common Details** presentation and the **Common Details – Promoted Properties** presentation. These two presentations serve as a base for understanding mediation primitives in general.

An overview of the function provided by the endpoint lookup primitive is presented, along with information about the primitive's use of terminals and its properties.

This primitive has a special relationship to the service message object (SMO). The presentation reviews those elements in the SMO context that are specifically used by endpoint lookup primitives.

In order to understand the endpoint lookup primitive, it is necessary to understand more than just the primitive's behavior and how it fits into a mediation flow. The endpoint lookup primitive interfaces with the WebSphere Service Registry and Repository, which is referred to as the registry in this presentation. Use of the registry is enabled through capabilities provided by the WebSphere Enterprise Service Bus and the WebSphere Process Server, which are referred to as the server in this presentation. You are going to learn about the service registry cache and administration of registries, capabilities that are provided by the server. You are also provided with references to documentation describing how to configure the registry.

The presentation then returns to looking specifically at the endpoint lookup primitive, covering error handling, problem determination and an example of its usage.

Overview of function

- Uses a registry to find service endpoints
  - Performs the lookup based on selection criteria
  - Initializes SMO with results for downstream use by the mediation flow
- Numerous criteria can be used for selection
  - Which registry to use for the lookup
    - Available registries are administratively defined within a WebSphere cell
  - Specifics of the requested service port type
    - Name
    - Namespace
    - Version
  - Associated classification, based on Web Ontology Language (OWL)
  - Associated properties and property values

The endpoint lookup primitive uses a registry to find service provider endpoints based on a set of selection criteria. The results of the lookup are reflected in the service message object which allows them to be used downstream in the mediation flow.

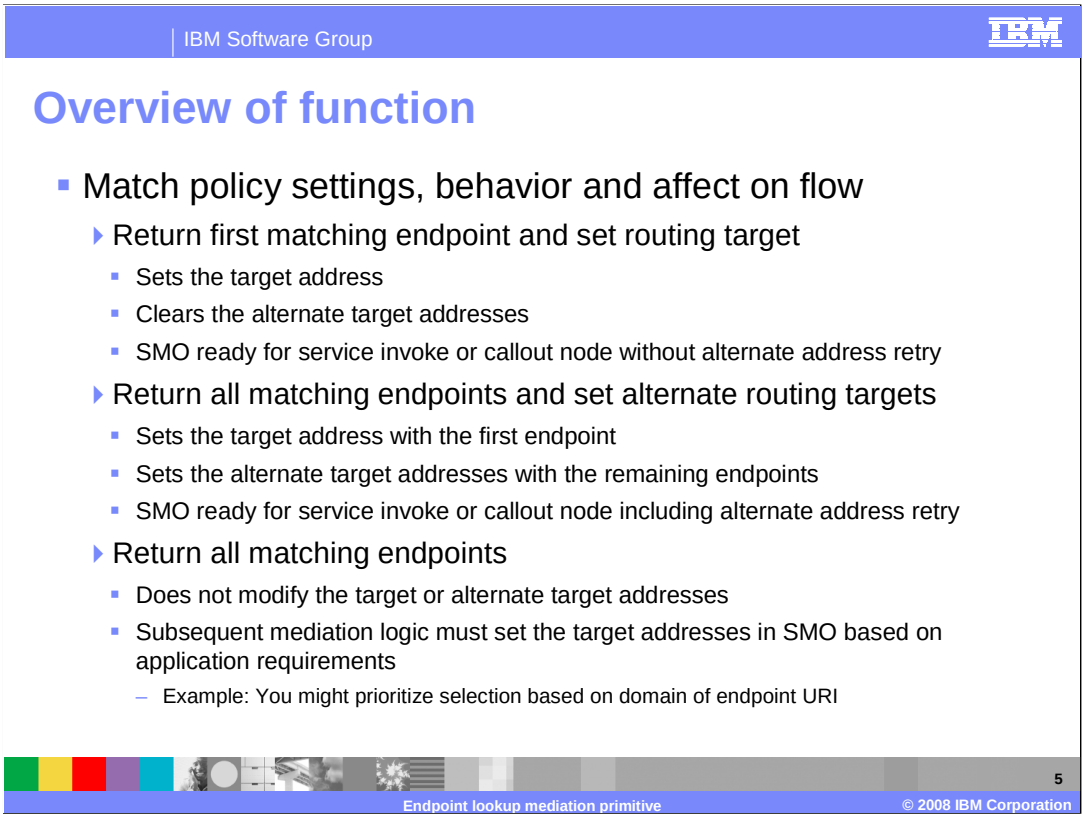
There are many different criteria that can be used for selection. First, multiple registries can be configured for use by the servers within a WebSphere cell and the endpoint lookup can specify which of these registries should be used. The service port type can be qualified based on name, namespace and version. The Web Ontology Language (OWL), provides a classification system which can be used as part of the selection criteria. Registered services can be associated with name value pairs which can also be used as part of the selection criteria. These selection criteria reflect the underlying capabilities of the WebSphere Service Registry and Repository.

Overview of function

- Context in SMO contains section for selected endpoints
  - List of endpoints selected, defining for each endpoint
    - Endpoint address
    - Properties and property values
    - OWL Classifications
    - Relationships to other registry entities
- SMO header set for dynamic endpoint usage
  - Setting of the target and alternate target addresses
  - Controlled by the match policy property
  - Affects what can be done in subsequent flow

The context section of the SMO contains a section in which the selected endpoints are placed by the endpoint lookup primitive. For each endpoint there is the endpoint address, a list of associated properties and property values, the OWL classifications and a list of relationships between the endpoint and other registry entries.

Service invoke primitives and callout nodes can use dynamic endpoints that are set in the SMO header. The setting of these fields, the target and alternate target address fields, is controlled by specification of the match policy property of the endpoint lookup primitive. Enabling the setting of these fields allows this primitive to initialize the SMO for dynamic endpoint usage by service invoke primitives and callout nodes.



The match policy setting determines how the target and alternate target address fields in the SMO header are affected by the endpoint lookup primitive. This has a resulting affect on what can be done downstream in the flow.

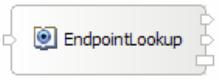
The first match policy setting is labeled return first matching endpoint and set routing target. This results in the target address field being set and causes the alternate target address field to be cleared of any addresses it might already contain. At this point, the SMO is ready to be used by a service invoke primitive or callout node configured for dynamic endpoints but is not ready for service call retry with alternate target addresses.

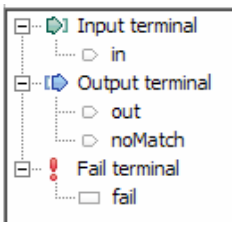
The next match policy setting is labeled return all matching endpoints and set alternate routing targets. This results in the target address field being set with the first endpoint returned and the alternate target addresses field being set with the other endpoints returned. At this point, the SMO is ready to be used by a service invoke primitive or callout node configured for dynamic endpoints and for service call retry with alternate target addresses.

Finally, there is the match policy setting labeled return all matching endpoints. In this case, the target address field and alternate target addresses field are not set. Therefore, using this match policy, downstream processing in the mediation must perform some logic that selects an endpoint and places its address into the target address field. The logic might also set other endpoints in the alternate target addresses field. What logic is performed depends upon your application requirements. An example of how this might be used is to look for an endpoint address that is within the same domain.

Terminals

- Terminals:
  - Input terminal
  - Two Output terminals
  - Fail terminal

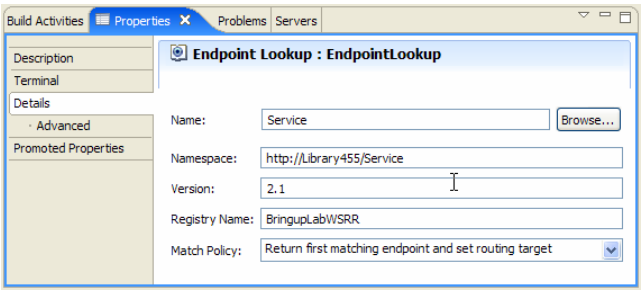




- Output terminals
  - out – if one or more matches is found the updated SMO is propagated
  - noMatch – if no matches are found, the unchanged SMO is propagated
- All terminals must be for the same message type

The endpoint lookup primitive has one input terminal, two output terminals and a fail terminal. There is one output terminal named **out** used when the endpoint lookup is successful and another output terminal named **noMatch** used when there was no service endpoint that satisfied the selection criteria. The output terminals must be for the same message type as the input terminal because the endpoint lookup primitive does not modify the message body structure. The slide shows an endpoint lookup primitive with its terminals and the terminals as seen in the properties view.

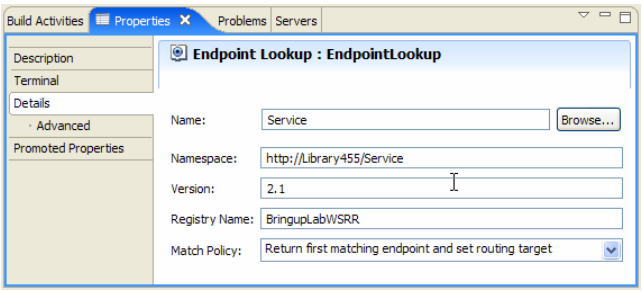
Properties



- Properties define search criteria for endpoint selection
- Name
  - Port type name
  - Typically matches the Interface on the Reference for the dynamic callout
- Namespace
  - Port type namespace
- Version
  - A freeform string used to represent the version of the port type
  - Matched with version field for port type in WebSphere Service Registry and Repository
  - There is no equivalent in WebSphere Integration Developer

The definition of search criteria for an endpoint lookup starts with the specification of the Port Type, which defines the interface the service endpoint is to support. The two values Name and Namespace define the definition of the interface. These normally are the same as the interface specified on the mediation flow component reference that is a associated with the dynamic callout node or a service invoke primitive in the flow. The Version is a freeform string which is meant to identify the version of the port type. It is matched with a version specification in WebSphere Service Registry and Repository. There is no equivalent concept of a version specification for interfaces in WebSphere Integration Developer.

Properties

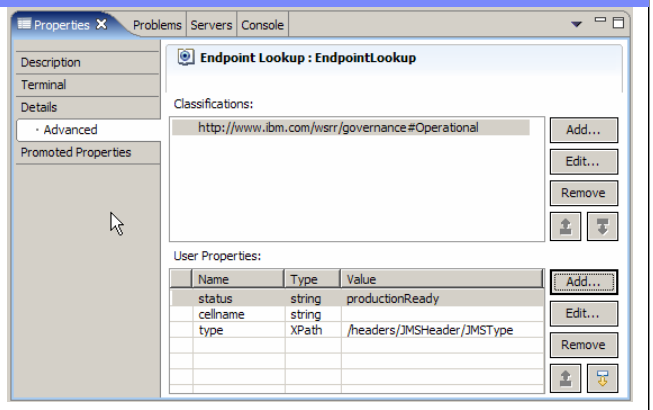


- Registry Name
  - Identifies the registry against which to do the lookup
  - Registries are administratively defined in the server runtime at the cell level
  - Leaving this blank results in use of the registry designated as the default
- Match Policy
  - Return first matching endpoint and set routing target
  - Return all matching endpoints and set alternate routing targets
  - Return all matching endpoints

The next property is the Registry name, which identifies the registry against which the lookup is to be done. The WebSphere Enterprise Service Bus and WebSphere Process Server manage registry references administratively, and this is the name the registry is known by in the server. The server designates one registry as the default, and that registry is used if this property is left blank. More details about how the server manages registry references is provided later in this presentation.

The Match Policy property can have any of the three possible settings shown in the slide. A full description of these match policy values and the behavior associated with them was provided earlier in this presentation.

Properties



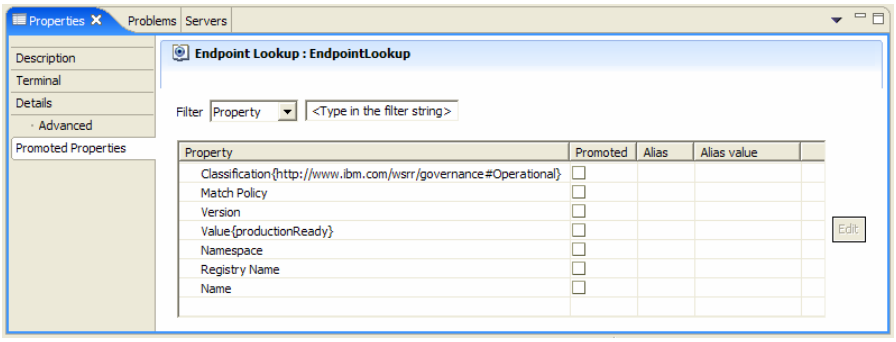
- Classifications
  - List of classifications associated with the selected endpoints
  - Each classification is specified as a URI as defined by the OWL classification system
- User Properties
  - List of user defined properties associated with the selected endpoints
  - String with value – selected endpoint must have specified property with specified value
  - String without value – selected endpoint must have specified property with any value
  - XPath – selected endpoint must have specified property with the value that was extracted from the leaf node element in the SMO defined by the XPath expression

The endpoint lookup primitive has an Advanced Details panel, as is shown here. It contains the Classifications property and the User Properties property.

The Classifications property is a list of classifications that should be associated with the selected endpoint. A classification is specified as a URI which is defined by the OWL classification system.

The User Properties property is a table of name value pairs that should be associated with the selected endpoint. The Name column contains the name of a user property. If the Type column contains XPath, then the Value column contains an XPath expression identifying an SMO element that contains the value for the user property. If the Type column contains string, then the Value column contains either the value for that user property or blank. When both Name and Value are present, the selected endpoint must have the specified user property with the specified value. When Value is left blank, then the selected endpoint must have the specified user property but it can have any value.

Promoted properties



- Promotable
  - Classification (individual rows)
  - Match Policy
  - Version
  - User Properties (the Value column)
  - Namespace
  - Registry Name
  - Name

All of the properties for the endpoint lookup are promotable.

The Classification property is a table with a single column. Individual rows in the table can be promoted.

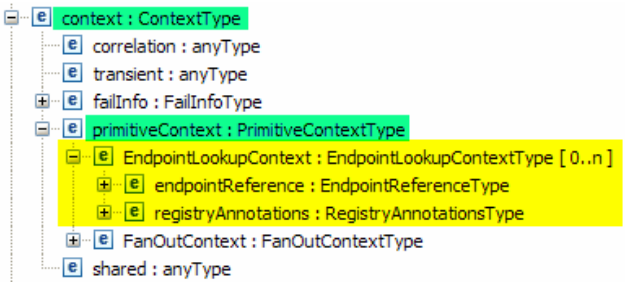
The User Properties table designates the Value column as promotable, allowing values for individual rows to be promoted.

Promoting the Registry name allows administrators to dynamically manage which registry is being used.

The remaining properties are all promotable, but promoting them and changing their values at runtime has to be done with great care. Changing the Port Type by changing the Name, Namespace or Version can result in a failure if the newly selected endpoints have an interface which is incompatible with the definition of the flow.

Changing the Match Policy typically implies that the mediation flow logic needs to be changed as well. Therefore, for most circumstances, this does not appear to be something that should be dynamically changed.

SMO usage



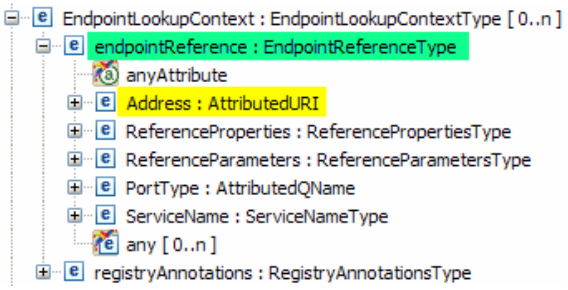
- /context/primitiveContext
  - Contains mediation primitive specific context information
- /context/primitiveContext/EndpointLookupContext
  - Array of endpoint information
  - Populated by endpoint lookup with result of registry lookup
  - For each endpoint in the array
    - endpointReference – defines the endpoint
    - registryAnnotations – additional registry information associated with the endpoint

The next few slides look at the SMO content that is specific to the endpoint lookup primitive.

Looking at the screen capture, you can see that within the context section of the SMO there is a primitiveContext section. Its purpose is to allow mediation primitive types to define a specific usage of the SMO which is unique to primitives of that type.

Looking again at the screen capture, you can see that the primitiveContext contains an EndpointLookupContext, which is an array of endpoint information. This array is populated by the endpoint lookup primitive with the results of a registry lookup. Each endpoint in the array has endpointReference data defining the endpoint and registyAnnotations data defining additional information about the endpoint from the registry. These are both covered in more detail on the upcoming slides.

SMO usage

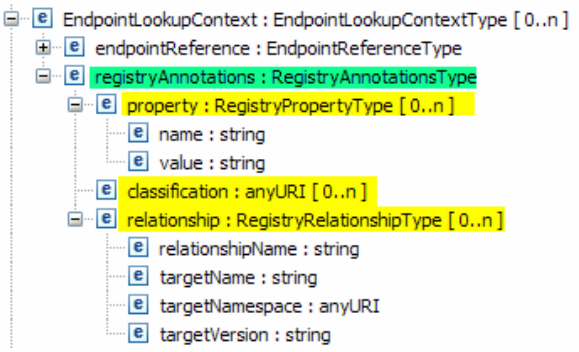


- endpointReference
  - Schema defined by the WS-Addressing specification
    - Schema → http://schemas.xmlsoap.org/ws/2004/08/addressing
    - Specification → http://www.w3.org/Submission/ws-addressing/
    - Use of this schema is consistent with:
      - Service Oriented Architecture (SOA) Core
      - Business Process Choreographer
      - WebSphere Application Server SPIs
  - Address
    - The URI needed to contact the endpoint
    - This value is used when setting the SMO header target address and alternate target address fields
  - Other fields – see ws-addressing specification for full details

The schema for the endpointReference is defined by the WS-Addressing specification as defined by the World Wide Web Consortium (W3C). Looking at the slide, you see the URLs identifying where to find this schema definition and specification. Using this schema provides consistency with the Service Oriented Architecture Core, Business Process Choreographer and WebSphere Application Server SPIs.

This presentation does not attempt to describe this schema. The key element from the schema to understand is the Address element. This contains the URI that is needed to contact the service endpoint. It is this value that is placed into the target address field or alternate target addresses field of the SMO header for use as a dynamic endpoint for a service invoke primitive or callout node.

SMO usage



- registryAnnotations
  - property
    - Array of name/value pairs
    - Arbitrary properties
  - classification
    - Array of URIs
    - OWL classifications associated with the endpoint
  - relationship
    - Array of relationships between this endpoint and other registry entities
    - Each relationship identified by a name and attributes of the related entity
      - relationshipName
      - targetName
      - targetNamespace
      - targetVersion

This slide examines the registryAnnotations, which is composed of three arrays.

The first is the property array that contains the name value pairs for the user properties associated with the endpoint.

The next is the classification array, containing the URIs of the OWL classifications associated with the endpoint.

Finally, there is the relationship array. It contains information about the relationship between this endpoint and other entities in the registry. Each relationship is defined by a relationship name and the name, namespace and version of the target entity.

Service registry cache

- Server runtime provides a cache for registry lookups
  - Intended to boost performance for registry lookups
    - Caching effective because registries are not normally subject to frequent updates
  - The cache is not exposed by the Mediation Flow editor
    - It is a hidden implementation detail to the Integration Developer
  - Server contains one cache per configured registry
    - Registries are administratively configured in the server
    - The presence of a cache is exposed to the administrator
  - The cache is populated “lazily” as lookups occur
  - A timeout period is associated with the cache
    - Individual cache entries are invalidated based on the timeout period
    - There is administrative capability to set the timeout value
    - Timeout value can be set to “0” to indicate not to cache entries

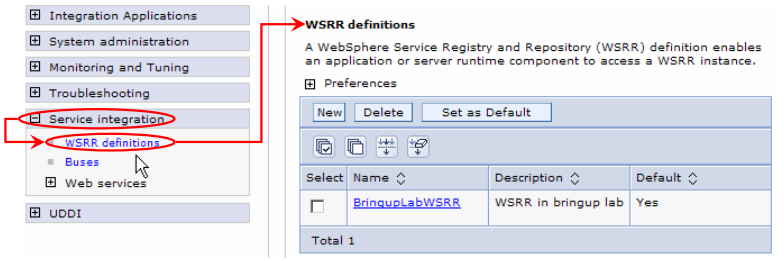
Performing frequent lookups to the WebSphere Service Registry and Repository can be expensive in terms of performance. Registries do not normally have frequent updates, so caching is a viable way to address this. Therefore, WebSphere Enterprise Service Bus and WebSphere Process Server provide a registry cache which is intended to boost the overall performance of registry lookups by reducing the number of calls made to the registry.

The endpoint lookup primitive and the mediation flow editor do not expose the registry cache. Therefore, from an integration developers perspective, the cache is hidden.

The cache is exposed to the server administrator who is responsible for configuring the registries. There is one registry cache per registry configured with the server. Population of the cache occurs in a lazy fashion, with entries being added as lookups occur. The administrator can assign a timeout period that is applied to cache entries, causing entries to be invalidated once they are older than the timeout period. If a timeout period is set to zero, the effective result is that caching does not take place.

Server administration of WebSphere Service Registry and Repository

- Service registries need to be configured
  - Endpoint lookup primitives must use configured registries
  - Registry definitions found in the administrative console:
    - **Service integration -> WSRR definitions**



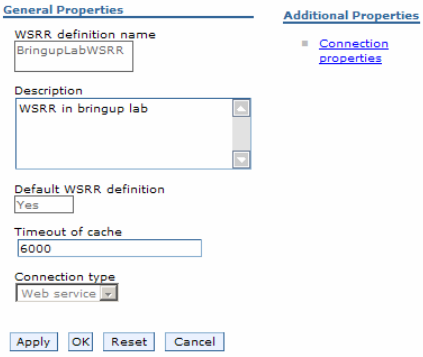
The next couple of slides show how the WebSphere Service Registry and Repository registries are administered in WebSphere Enterprise Service Bus and WebSphere Process Server.

Note that in order for an endpoint lookup primitive to use a registry, the registry must be administratively defined as described here.

This slide shows the navigation in the administrative console. On the left panel, open Service integration and then select WSRR definitions. This opens a panel containing a list of the configured registries. Clicking on one of the registries causes it to open in a registry properties panel, as shown on the next slide.

Server administration

- Registry definitions maintained at cell scope
- WebSphere Services Registry and Repository definition name
  - Used to identify this registry instance
  - Endpoint lookup configuration uses this value to identify the registry
- Default WebSphere Services Registry and Repository definition
  - Designates this registry as the default
  - Leaving registry name blank in endpoint lookup causes this registry to be used
- Timeout of cache
  - How long cache entries remain valid
  - Value of 0 prevents caching



This slide shows the properties that are used to configure a WebSphere Service Registry and Repository. Within a WebSphere cell, these definitions are maintained only at a cell scope, and cannot be specified at the node or server scope.

The first field is the ‘WSRR definition name’ field. This property is a name used to identify this registry instance. This is the value that you specify in the endpoint lookup primitive to identify the registry to use for the lookup.

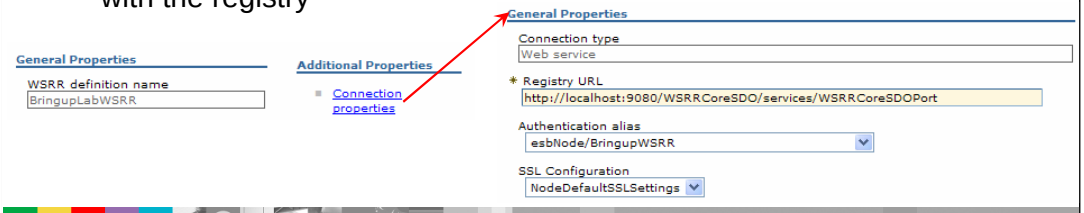
The Description property provides a text comment describing this registry.

The ‘Default WSRR definition’ field, if set to yes, designates this registry instance as the default registry. The default registry is the one that is used when the registry name property in the endpoint lookup is left blank.

The Timeout of cache property specifies how long cached entries remain valid. A value of zero prevents any caching of entries for this registry instance.

Server administration

- Connection type
  - Currently Web service is the only allowed choice
- Registry URL
  - URL to connect to registry instance
- Authentication alias
  - Defines user ID and password needed to authenticate with the registry
- SSL Configuration
  - Defines the SSL configuration in the server needed to communicate with the registry



From the panel shown on the previous slide, clicking on the Connection properties link allows you to specify these properties for the connection to the WebSphere Service Registry and Repository.

The Connection type defines what type of protocol is used to connect with the registry. Currently, Web service is the only connection type.

The Registry URL property provides the specific URL needed to connect to this registry instance.

The Authentication alias property identifies an authentication alias containing the user ID and password needed to authenticate with the registry.

The SSL Configuration property identifies the SSL configuration definition needed to connect with the repository.

Configuration of the WebSphere Service Registry and Repository

- Services must be correctly defined in the registry
  - This presentation does not address how this is done
- See the WebSphere Service Registry and Repository Information Center for complete information
  - http://publib.boulder.ibm.com/infocenter/sr/v6r0/index.jsp
  - See topic: **Loading SCA module definitions**

Describing how to configure and administer the WebSphere Service Registry and Repository is beyond the scope of this presentation. However, it is important that the services be correctly defined in the registry if the endpoint lookup requests are to result in matches.

Information on configuration of the WebSphere Service Registry and Repository is provided in the information center at the URL shown on this slide. For initializing the registry with service endpoints defined by SCA modules, see the topic entitled Loading SCA module definitions.

Error processing

- MediationBusinessException (fail terminal flow)
  - Registry not currently available
  - Configured URL for registry is incorrect
    - Correct syntax but wrong, such as an incorrect port specified
    - Incorrect syntax (malformed URL)
  - Registry name not found
    - Incorrect name specified in mediation primitive
    - No administrative entry in the server for the for specified registry
  - User Property XPath expression problem
    - Valid XPath but element does not exist in the SMO
    - Invalidly formed XPath expression

There are several conditions that cause a MediationBusinessException to occur. When any of these happen, the mediation flow continues through the fail terminal if it is wired, otherwise the exception is re-thrown and the mediation flow is ended.

One issue is that the registry specified is not currently available and therefore cannot be contacted with the lookup request.

Another possible reason is if the URL for the registry has been incorrectly specified in the administrative definition of the registry. This can be a URL with a good syntax but has a mistake such as an incorrect host or port specification. It can also be for a malformed URL.

The MediationBusinessException can also occur if the administrative definition for the registry specified in the endpoint lookup properties cannot be found. This can occur if the administrative name for the registry is misspelled, or if the registry was never administratively defined.

Another reason for this exception is if the User Properties table contained an XPath expression which can not be resolved. This can occur if the XPath expression is incorrect or if this instance of the SMO does not happen to have the element defined by the XPath.

Problem determination

- Call to the registry fails
  - Ensure the registry service is running
  - Ensure registry is configured correctly in the server
  - Ensure mediation primitive has specified the right registry entry
- Not getting expected services returned from registry
  - Ensure selection properties configured on primitive are correct
  - Ensure correct WSDLs and XSDs have been loaded into the registry
  - Ensure your primitive is configured for the right registry
  - Consider possibility that a cached entry is being returned
    - Result does not reflect recent updates to the registry
    - When doing development or test, it is be best to run with cache timeout = 0
- Target address not set in SMO
  - Check logic for setting the target address

Several things can go wrong at runtime that are inherent in the complexity of the interactions involved. There is interaction between the endpoint lookup primitive, the server administration of registries, and the call to the registry. This slide gives you some things to look for when your endpoint lookup primitive fails at runtime.

The first set of issues revolve around a failing call to the registry. If this is the case, ensure that the registry is running and that the configuration of the registry in the server is correct. Also check that the mediation primitive has specified the correct administrative name for the registry to be used.

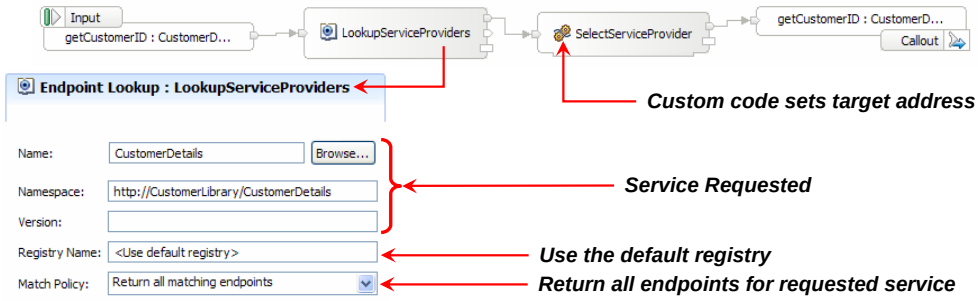
The next set of issues revolve around the results from a lookup apparently not returning the expected services. The first thing to check is the configuration of the endpoint lookup primitive to ensure that the various search criteria properties are correctly specified. If so, then make sure the correct XSD and WSDL files have been loaded into the registry. And finally, make sure that the endpoint lookup is configured to use the right registry.

If there have been any registry updates, consider the possibility that your endpoint lookup is being resolved through the registry cache rather than a call to the registry. This is not likely to be an issue in a production environment, but is definitely an issue during development and test. As a best practice, use a cache timeout of zero to prevent caching during the development phase and possibly also during the test phase.

If the dynamic callout is failing because the target address is not set in the SMO, there are a couple of things you can check. If the endpoint lookup is configured to have a Match Policy of one, check to see why no endpoint was returned from the lookup. If the endpoint lookup is configured to have a Match Policy of all and one or more endpoints was returned, check the mediation flow logic following the endpoint lookup to determine why the target address was not set.

Example

- Select endpoint by domain
  - Endpoint lookup finds all endpoints for a service
  - Custom code then:
    - Examines the endpoints returned
    - Selects an endpoint based on domain
    - Selection favors providers in the same domain



In this example, the selection of the endpoint is optimized to use a service provider in the same domain as the mediation. In the screen capture, you can see the mediation flow and the properties for the endpoint lookup. Notice that the lookup uses the default registry configured in the server and the match policy results in all endpoints for this service being returned. Following the endpoint lookup is a custom mediation primitive. It examines the address URIs of the returned endpoints looking for one that is in the same domain. If found, it places that into the target address element of the SMOHeader. If none are in the same domain, one endpoint is arbitrarily picked.

Summary

- Examined the endpoint lookup mediation primitive


Endpoint lookup

- Overview of function
- Use of terminals
- Definition of properties
- Service message object (SMO) usage
- Service registry cache
- Server administration of the registry
- Configuration of the registry
- Error handling
- Problem determination
- Example usage

In summary, this presentation introduced you to an overview of the function provided by the endpoint lookup primitive, along with information about the primitive's use of terminals and its properties.

Since this primitive has a special relationship to the service message object elements in the SMO context, the schema for this was examined.

In order to understand the endpoint lookup primitive, it is necessary to understand more than the behavior of the primitive itself and how it fits into a mediation flow. The endpoint lookup primitive interfaces with the WebSphere Service Registry and Repository. Use of the registry is enabled through capabilities provided by the WebSphere Enterprise Service Bus and the WebSphere Process Server. You learned about the service registry cache and the administration of registries that is provided by the server. You were also provided with references to documentation describing how to configure the registry.

The presentation then returned to looking specifically at the primitive, covering error handling, problem determination and an example of its usage.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WBIV61_EndpointLookupPrimitive.ppt

This module is also available in PDF format at: ../WBIV61_EndpointLookupPrimitive.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere

A current list of other IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.